UNIVERSITY OF HERTFORDSHIRE

School of Physics, Engineering and Computer Science

MSc Artificial Intelligence & Robotics

7COM1039-0109-2021 - Advanced Computer Science Masters Project

Detecting vehicles from low resolution images using deep learning models.

ABSTRACT

Due to increase on numbers of vehicle, in the traffic management it is very essential task to detect vehicles in order to minimize accidents and casualties. Therefore, government has installed many devices like CCTV in smart cities. But in some adverse conditions, devices capture noise along with the image or in some cases due to lack of equipment in some areas, it is very essential to zoom in and detect vehicle. But due to low resolutions, detection is hindered.

Therefore, in this paper I have proposed methodology to mitigate these situations. In my proposed methodology, I used R-ESRGAN for reconstructing images in super resolution and State of art detector YOLOV4 for detecting both vehicle and types of vehicles. Started with research question as derivatives to answer, I have executed my hypothesis.

Although results were not satisfactory due to latest pre-trained R-ESGAN model, but it was concluded that my proposed methodology works in order to mitigate problem.

Keywords: Traffic management, car detection, Low resolution images, Enhanced Super resolution GAN(ESRGAN), You Only Look Once (YOLO).

Acknowledgement:

I would want to express my gratitude to Mr. Raimund Kirner, who not only assisted me in getting a handle on this project but also provided me the incredible opportunity to work on it. I'd also like to take this opportunity to thank Dr. John Noll and Dr. Catherine Menon, two of my professors who helped me become more knowledgeable in research methodology.

I must also express my gratitude to both my parents and God for the incredible support and assistance they provided over the duration of this project. It would have been extremely challenging to complete our assignment without their assistance. All authors mentioned in the proposed methodology are considered as equal contributor to this project.

Due to the fact that I am still relatively inexperienced in this study field, I would like to thank every author that was cited in this work.

Contents

Introduction:

The overall number of vehicles and population worldwide have increased significantly over the past few decades, which has increased traffic and caused a spike in accidents, delays, and air quality. Many smart cities or urban areas throughout the globe have technologies like CCTV cameras, sirens for emergencies, pols for power, etc. built and installed by the government to maintain netizens' safety and a hassle-free life. The traffic management department is one of them.

The primary objectives of traffic management are to ensure the safety of road users and to keep traffic flowing. One of the key components of traffic management is vehicle detection. Knowing how many vehicles are on the road at any given time helps to ensure that traffic flows smoothly and this data is then processed to help identify potential accidents and jams. Effective traffic management is within the domain of computer vision. And in recent years, vehicle detection algorithms have been evolving and are generating more accurate and quick conclusions.

Traffic cameras are usually high resolution and can provide good image quality even at night or in low light conditions.

Similar to how a human new born brain develops object awareness by seeing patterns, linking all the knowledge that has been learned, and identifying it with a specific item, deep learning techniques uses artificial neural networks mimicking human brain that include an at-least one hidden layer. They have the ability to automatically execute feature detection from vast quantities of labelled training data. Deep learning techniques make use of neural network algorithms. Initially, these models are trained with dataset containing images or numbers of frames of a video.

After training, these models are tested on different datasets in order to record their accuracies.

In other words, vehicle detection is to determine and pinpoint location of an object in a particular frame or image. (Maity, Banerjee and Sinha Chaudhuri, 2021)" *Deep learning methods have an innate feature extraction capability, which makes them far more appealing to researchers than traditional approaches because it greatly reduces classification job failures that are caused by inaccurate manual feature extraction.*" [19] and according to Ambica [5]there are many different methods for performing vehicle detection and most commonly used are like CNN based namely Alexnet introduced by Krizhevsky, Sutskever and Hinton,in 2012[16],later advancing by other researchers who proposed versions like Fast Region-Based Convolutional Network method(Fast R-CNN), algorithm utilises the Region Proposal Network (RPN), Faster R-CNN, Region-based Convolutional Neural Networks (R-CNN) & Region-based Fully Convolutional Network (R-FCN) and Single Shot Detector (SSD) ,YOLO (You Only Look Once),etc. And these models have gained popularity due its high detection rate. Among these, YOLO is state of art detector used and has one of the best accuracies.

Research Question

However, Pang et al. (2019) [21] argues that the time and memory required for recognition in traffic management have grown, making it too slow and unsuitable for use with current technology. And they are not perfect under all circumstances and can be fooled by objects that resemble vehicles, such as bushes or shadows or unique backgrounds that develop in a real setting may yield more false positives, such as image of a dark colour vehicle in night or white automobile in sunny day or in snow. And in some scenarios, due to enlarging the frame(image) in order to detect vehicle/s pixels degrades further which disturbs the detection. Furthermore, performance drops quickly, as resolutions are dropped also. This phenomenon is just like as few humans who suffers from iris focal problem and need spectacles to mitigate.

For instance, it would be easier and more efficient for vehicle detections algorithms to detect vehicle high resolution image from fig.1 than detecting vehicle from low resolution images like fig.2.

Even though it's clear in human vision but difficult for networks to recognise and it is also very difficult to fully solve this issue due to the limited number of images and datasets available for testing and training for both machine learning and deep learning approaches.

Fig.1

Fig.2

And another issue faced is when data (images or video) is transferred from devices to Traffic management, it has been observed that noises and distortion are arise, which makes difficult to remove it. As a result, new methods like satellite imaginary for traffic management are needed that can handle the increased demand for speed and accuracy which tends to be expensive.

Problem statement:

In recent years few researchers have try and successfully solved this problem, and in this report, we will try with a simple yet efficient approach. There are 2 ways to overcome this problem 1) To train object detection model for low resolutions images but model would be unable to perform detection with high resolution data.2) To address our problem of detecting automobiles/vehicles from low-resolution picture datasets, I propose a multistage procedure for generating high-resolution images from low-resolution pictures using Enhanced super resolution GAN(SRGAN) and then

further identifying cars/vehicles from the generated high-res images using YOLOv4(a version of YOLO).

In any research problem, it is essential to first identify the research questions. These questions will guide the proposed method and help to ensure robustness and accuracy. For this problem, I have formulated the following research questions:

Research questions:

1. Can Real-ESRGAN generate realistic image with optimized noise?
2. Can generated super resolution dataset from Real-Enhanced Super Resolution Generative Adversarial Network model (ESRGAN) to assist YOLO model in vehicle detection?
3. What is mAP (unit of accuracy) of trained weights of YOLOv4?

Related works:

In recent years, many algorithms have been developed by researchers to solve different problems in the different field of computer vision. Machine learning and deep learning approaches have been found to be very effective in this application. In this section, we will outline the journey of the different techniques used by the researchers in recent years to overcome this issue. We will also discuss the related work in this field so that the reader can get a better understanding of the topic. Finally, we will study the recent works in deep learning approach to this field of computer vision and its potential applications in future work section.

Related work based upon method of recognising from Low resolution/Pixels

Few researchers like (Bautista et al., 2016) [1] have solved this problem using Convo lutional Neural Network (CNN), and as Bautista and his team believed CNN is a type of machine learning algorithm that is based on the human nervous system, "*It can lea rn how to recognize objects that do not change, no matter what perspective viewed fr om*". So, in order to assess the effectiveness of the CNN-based system in the identifi cation and categorization of vehicles, he and his team conducted four tests utilising fr ames from low quality cameras. The size and quantity of filters were replicated upon i n the first and second tests, respectively. Thus, to improve the performance of the ne twork, the third and fourth experiments investigated the idea of integrating deep learn ing into the feature extractor layer and classifier layer, respectively. The results show ed that the CNN-based system was up to 96.47% effective in the identification and d etection of vehicles. Though this proposed methodology is effective, time and comput ational power required was high, it would have been beneficial to upgrade technology in realm of traffic management. While Tas et al., 2022[29] proposed VGG16 model th at built along with a CNN model built from scratch. The accuracy of these predictions was evaluated against new data sets containing tiny and low-quality vehicle images, which showed that it performed best by another pre trained version created by Finetu ning (which performed slightly better than 96%) (Note: So far, this is latest piece of re search in this field and problem). Due necessity, detecting vehicle from video or in th e real time this approach would not solve our problem statement entirely.

Where else, many research paper and studies were conducted with dataset of aerial images, for instance, in 2016 group of researchers Cao, Wang and Li[Ca], to overcome barrier of limited dataset available and training detector in satellite imaginary field, They used a super resolution framework, a sparse-coding based reconstruction algorithm and a robust vehicle detection via linear SVM based search to convert the detection performance from low-resolution satellite image field, enhancing the discriminativeness of the transferred patches in high-resolution for the purpose of training vehicle detectors and a linear "*SVM-based search that allows massive parallel* " processing respectively. Furthermore, following same pattern of removing background, Chen et al., 2021 [4] proposed background elimination, a background subtraction approach which recognises moving objects by first creating a background picture from the multi-temporal images and then eliminating it. Thus, this mythology would be ineffective to recognise vehicle (in deep frame) in severe weather conditions.

For example.

With comparisons to a number of existing and alternative methodologies mentioned in their scientific article and above mentioned, [23] Rabbi et al., 2020 presented an end-to-end system that uses LR satellite imagery to detect objects. SR and detector networks are in their architecture. Utilizing two separate datasets, they compared AP detection values using different SR methods and detectors. The suggested SR network with faster R-CNN performed best for small objects in satellite imagery, thoroughly tested their strategy. In spite-of, their reported better performance, their proposed method would be more accurate on labelling from satellite imaginary field OR one of the best approaches would be by removing noise and and shuffle pixels and reconstruct from low resolution satellite imaginary.

While on other hand, researchers like Ji et al., 2019[11] and his team, proposed CycleGAN based SRCNN (Super Resolution CNN) and they termed it as "CycGANSR". They conduct their experiment using the DLR Munich data collection, which was collected utilising a DLR 3K camera system around the city of Munich. It has 20 photos with a resolution of 5616 by 3744 pixels and a ground sampling distance of around 13 cm (GSD). The UCAS-AOD data collection, which had 510 satellite photos with a resolution of 659 x 1280 but no GSD information, was another dataset that was utilised. Although, this piece of research was based on dataset with satellite

images, their methods proved to bypass the requirement of dataset of low/high images. Although [23] and [11] both proposed better strategy but computation cost and cost of tools required is too high.

Recent work based on Object detection

While some research related to object detection for low resolution images were conducted to solve problem of detection from low resolution, for instance, Peng et al., 2014[22] in order to solve problem of recognising logos from low resolution images, they proposed new Vehicle logo recognition method using statistical random sparse distribution (SRSD) feature and multiscale scanning. And as this method used correlations of global pixels pairs as techniques that allows a system(network) to automatically discover the representations(regions) needed for feature detection or classification from raw data and results were most accurate in daytime. Although this method outperformed all traditional methods in the field VLR (as per 2014) with accuracy of 97.21% but in our case, even if we train network to detect vehicles (in distance) network the network lacks depth to recognise all the objects in the frame.

While many research studies mentioned, indirectly answer our problem statement, Kim et al., 2019 [15]performed experiment using dataset of images with following criteria,

- Near set of 100 images with ~ 10 m distance and Width & height > 130
- Intermediate set of 100 images with distance of 13 to 17 m and Width & height between 70-100
- Far set of 100 images with distance of 20 m and Width & height less than 70

Their proposed methodology was to use Super Resolution GAN(SRGAN) for uplifting resolution and Convolutional Neural Network (CNN) for recognizing object, as a result average accuracy was increased from 17.3% to 68%.

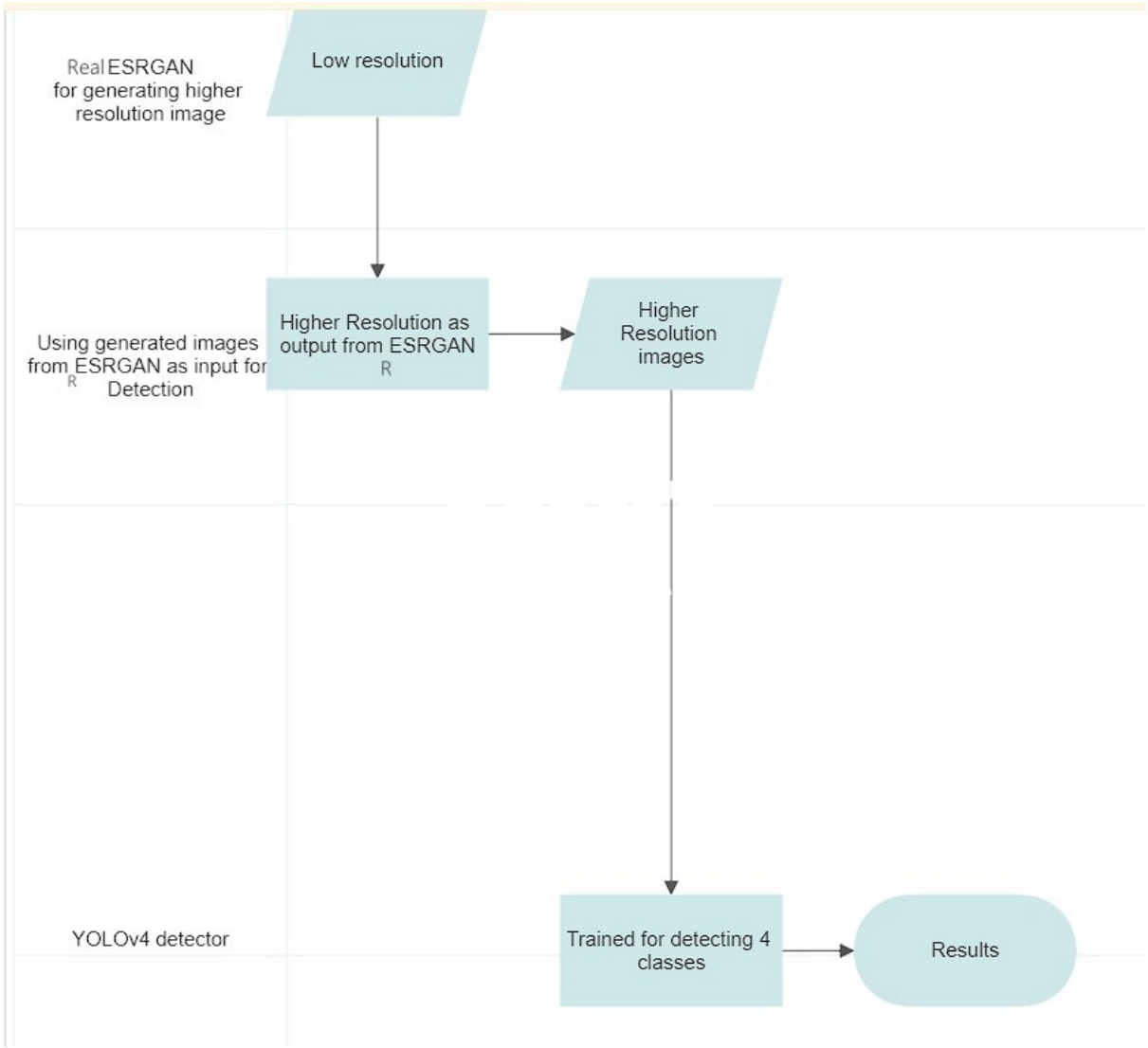
While Lam, Ng and Chan, 2019[La] came up with median filtering to remove noise fro m images using multiple IOU(further in YOLO) mIOU calculation, although there was no significant difference in precision rate but there was increase of recall rate for loca tion #53(a location number in their paper) but it completely diverse for location #76 (a location number in their paper) which conclude that this proposed methodology is q uick but is depended on frame resolution. Therefore, due these contradicting results we cannot use this methodology.

Proposed Methodology:

In above section Related works, we explored recent works, but most of these works are based on aerial imaginary, so even if these methods used cannot be used on our dataset and our problem, as the perspective(object) changes.

Thus, combining all the information gained and taking inspiration from [Ki], [23], [Gi_1] and other researchers mentioned in the report, I believe proposed methodology of using ESRGAN (Enhanced Super Resolutions GAN) and YOLOv4 (because of BOF {bag of freebies for post processing in order to improve accuracy} and BOS {bag of freebies for Data Augmentation} would work. And in this section, I will provide practical and theoretical proof for selecting models used.

To my best knowledge, this work is one of few research done to solve problem of recognition of vehicles from low resolutions. And I have used my best knowledge and executed proposed methodology with free of cost. Although, I strongly believe, use of proposed methodology in the real time other than detecting vehicle could be worse.



Ideology behind, is that first to transform images from low resolution or with noise like fog using pretrained R-ESRGAN model, so that images generated are

reconstructed lesser noise around the object (vehicle), then use Yolov4 trained for detecting 4 classes (for faster training and detections).

I have following two things to accomplish in this report.

- Verify whether if R-ESRGAN is assisting or not. And show comparison of detection using my trained YOLOv4 with and without R-ESRGAN like [Ki].
- Derive accuracy of YOLOv4, I trained. And show, samples of detection on low resolution and detection on high resolution (not from R-ESRGAN) images.

Datasets:

While working with deep learning models, image dataset is very essential part for both training and testing. In this project, I have used Datasets that are available on public sources.

Dataset_1: In this dataset, I collected few images from https://www.google.com/ with different dimensions and resolutions. I intended to use this dataset for verifying proposed methodology.

In this dataset, I collected 12 images with resolution of lower than 64x61 (cropped from frame) and 3 of each class. Apart from that, in this dataset, I have downloaded 2 images from bridge view (of city Delhi, India) with dimension of 125x71.

Note: None of the images from Dataset_1 was used in the Training process.

Dataset_yolov4: This dataset contains, all types of vehicles in United Kingdom and its free for public use. This dataset provides with “Test” and “Train” sets. Although, we would not be using in this project. So, randomly selected 1111 images with different numbers of vehicle and different classes.

I have selected 4 classes, 1) Ambulance, 2) Truck,3) car and 4) bus for training and detections as I believed we need only four classes to detect in for basic Traffic Management operations. And training less class can reduce time.

Real-ESRGAN for generating realistic image:

As first stage of problem statement is to generate images with good textures and details so that performance of detection is not hindered. Therefore, in this section we will go through different methods of generating super resolution images and prove my choice of selecting pre-trained Real-ESRGAN model.

Super Resolution:

This is the process of reconstructing higher resolution images from lower resolution images. Deep learning techniques used for super resolution have widen their horizon

(and still is) in the various fields like astrophysics, astronomy, medical and many more. As our problem statement deal with problem of low resolution and also zooming-in on the object (in the image), so we will go through super resolutions methods that can be used to mitigate our scenario.

Metric score:

Peak signal-to-noise ratio (PSNR) is an expression for the ratio of a signal's peak permissible value (power) to the power of deformed noise that up/degrades the quality of its representations. In other words, according to research article by Nadipally, 2019) [20], the PSNR determines the PSNR ratio between image before upscaling or downscaling and after upscaling or downscaling image. In following image, mathematical equations are shown.

$$PSNR = 20\log_{10}\left(\frac{MAX_f}{\sqrt{MSE}}\right)$$

Figure 1 - Peak Signal-to-Noise Equation

where the **MSE** (*Mean Squared Error*) is:

$$MSE = \frac{1}{mn}\sum_{0}^{m-1}\sum_{0}^{n-1}\|f(i,j) - g(i,j)\|^2$$

Figure 2 - Mean Squared Error Equation

Source: https://www.ni.com/en-gb/innovations/white-papers/11/peak-signal-to-noise-ratio-as-an-image-quality-metric.html

Where, f depicts the original image's matrix data.

g: is The matrix dimension of an enhanced image(in our case)

m: the number of pixel rows in the image.

i: row's index.

n: The number of pixels in the image's columns

j: column's index.

MAX_f: The highest signal value in the image.

"Deep learning-based SR models have been intensively researched in recent years due to the improvement of deep learning techniques, and frequently attain state-of-the-art performance on many SR benchmarks" according to (Khandelwal, 2021) [14].

Interpolation based methods:

Traditional methods of upscaling were based upon enlarging image and then fill in the blanks with appropriate texture. These methods include nearest-neighbour interpolation, linear, bilinear, bicubic interpolation, etc.

The nearest-neighbour interpolation: This is a simple approach, without taking any of other pixels into account, it chooses the value of the closest pixel for each area that needs to be interpolated.

Bilinear Interpolation: This technique initially applies linear interpolation to x-axis of the picture before moving on to y-axis or vice versa. It performs significantly better than nearest-neighbour interpolation while been quick, since it produces a quadratic interpolation with a receptive field of size 2x2.

Bicubic interpolation: While Bicubic interpolation takes receptive field of 4x4 in account while upscaling image.

With knowledge of these basics, it would be easier to understand methods and approach mentioned below and execute one of them and select.

SRCNN:

In 2014, Dong et al., n.d [7] proposed of using Convolutional networks layers for upscaling image, hence the term “SR-CNN”. It has three CN layers 9x9 for patch extraction ,5x5 for non-linear mapping and 5x5 for reconstruction. The model take image with high dimension but with less pixel density, then doing Bicubic interpolation, first layer of CN will extract patches (newly formed) and a feature map is created (for LR) then whole frame/image (not complete image) is scanned and another feature map(for HR) is created and finally last layers fill with solid textures where needed(as original input has less pixel density). Although results proved effectiveness of the proposed method, but this model is highly concentrated on the minimizing MSE.

And as sharp edges (sometimes clustered edges) in the image would have insignificant factor in human perspective but it is not same for PSNR metric. And training this model is complex.

Perceptual loss:

Rather than focusing on the pixel loss and reconstructing image, in 2016 [13] Johnson, Alahi and Fei-Fei, n.d based on the idea of training a neural network (VGG16) for image classification, so that network can learn textures and dimensions of object (patterns), they proposed Perceptual Loss method for obtaining super resolution images. In the nutshell, taking set of higher resolution image and low-resolution image as an input and rather than comparing mean square error, with the help of VGG-16 network their features are compare and image is reconstructed, the loss while reconstruction is called “feature reconstruction loss” and model is trained to minimize(optimize) this loss while image processing. But this method lacks network’s depth thus generating extra grained details therefore it cannot be used in our methodology.

Similar in same year, Shi et al., n.d. [28] proposed Sub-pixel Convolution method for super resolution, in this method instead of bicubic interpolation, using convolution method the input image will go directly from one hidden layer to another while preserving pixel. So, each new would have less pixel to process which makes it faster. Then in last using sub-pixel convolutional layers (channels which are red, blue and green) and image is reconstructed with “pixel shuffle” by assigning (or arrange) on the original image.

However, these methods require high computational power and sometimes results have extra layer/s of region feature(patches), so in our case, there is a risk of developing extra feature which might change the shape of vehicle making detection difficult.

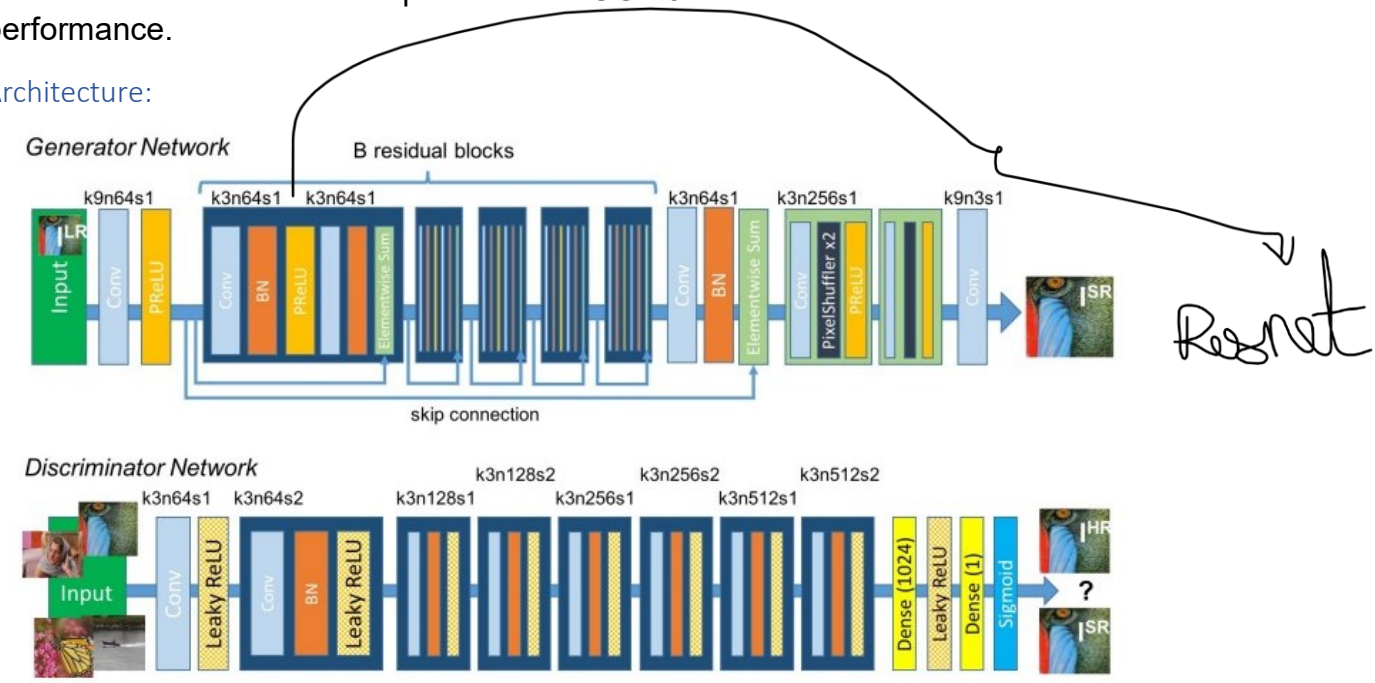
SR-ResNet:

In the same year (2016), Ledig et al., [17] proposed combination of SR-CNN and ResNet based upscaling method, as they believed “*High upscaling factors, for which texture detail in the reconstructed SR images is often missing*”. So, in order to optimize MSE, they proposed deeper network with 16 Residual blocks, skips connections, parametric P-ReLU (learnable) and lastly Sub-pixel convolutional.

SR- Generative Adversarial Networks

In the same article, [17] proposed Generative Adversarial Network (GAN) based upscaling method to capture the natural image manifold, in which they used VGGNet (as it is very useful for classification)., They also suggested deriving a hybrid loss by combining the adversarial and content losses. They also developed an improved content loss to compare more high-level aspects of the image by examining intermediate activation of the pre-trained VGG-19 network in order to further boost performance.

Architecture:



Developed by Ian J. Goodfellow in 2014, generative adversarial networks (GAN) are a novelty in machine learning. GANs can be trained to produce data that is realistic (with good opinion score). In a summary as shown inn above figure, GAN is made up of two neural networks: the generator and the discriminator. The generator generates new data samples, while the discriminator determines whether the data is real or fake.

The generator network is made up of a sub-pixel (Pixel Shuffle which upscales image by 2) convolutional approach and 16 residual blocks (in the original paper) that modify the image at a relatively small level. Each residual block for the generator consists of two sets of the conventional (Conv-BN-PReLU) block, with a skip connection coming after each set of the conventional (Conv-BN-PReLU) block and a constant channel number of 64 for all blocks. Hence, when image is upscaled (reconstructed) image looks realistic (with natural looking features).

Where else, the discriminator network is made-up of (Conv, BN and leaky ReLU) block and later connected with block consist of Dense layers, Leaky ReLU and final connected to block of 1 Dense layer as Discriminator is binary classifier and sigmoid function for deciding whether data is “real or fake”.

In practical, in this (every GAN) Discriminator Network is trained directly to detect realistic data and Generator is trained indirectly through Discriminator to generate more realistic data to fool discriminator.

Loss Function:

The weighted sum of a content loss (based upon VGG-19) and an adversarial loss component is referred to as perceptual loss (l^{SR}. And as VGGNet is pretrained therefore it has fixed loss where else, Adversarial loss (GAN loss) can be optimized as per requirement. Adversarial loss is loss

$$l^{SR} = \underbrace{\underbrace{l_{\mathrm{X}}^{SR}}_{\text{content loss}} + \underbrace{10^{-3} l_{Gen}^{SR}}_{\text{adversarial loss}}}_{\text{perceptual loss (for VGG based content losses)}}$$

Source: https://arxiv.org/pdf/1609.04802.pdf

In the experiment conducted at London, they human percipients to score (from 1-5) data generated and they concluded that data were SRGAN had generated more realistic images with average score of around 4.5.

Note: Few methods also often generated images which were off-balance colour (off texture), and a human eye would recognise this. So, keeping human perspective in the mind, a new scoring method “Opinion scoring” was required for assessing upscaled images, in this method researchers asks for human volunteers for scoring upscaled images (and many more image processing) in various ways (With their consent).

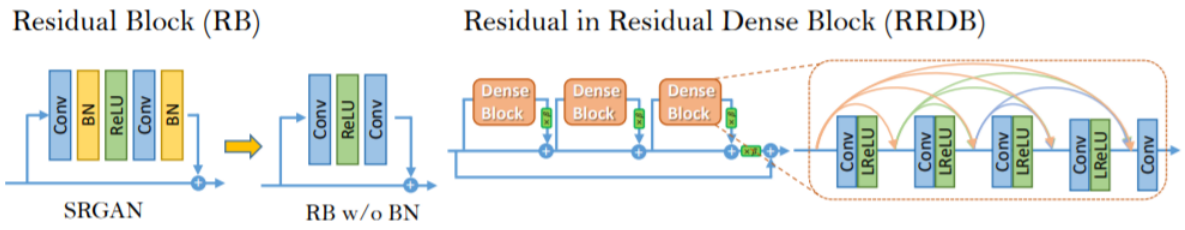
Enhanced SRGAN (E-SRGAN):

As title depicts, with three enhancements in SRGAN, in 2018 Xintao Wang et al. [30] proposed Enhanced SRGAN in order to further enhanced visual quality generated data from SRGAN by combining the architecture of SRResNet and SRGAN.

Enhanced Network architecture: They introduced Residual in Residual Dense block (RRDB) to increase size of the network and making easier to train. Thus, two changes are made to the network design in particular to improve the quality of the generated image of SRGAN:

- Removing all layers affected by Batch Normalization (BN)
- And RRDB to replace them with dense block.

As shown, in figure below.



Source: https://arxiv.org/pdf/1809.00219.pdf

Therefore, removal of BN layers results in a gain in performance as well as a reduction in the complexity of computation and the amount of memory that would be used.

When compared to the initial residual block in SRGAN, the structure that is produced by RRDB for the Generator Network is significantly deeper and more complicated, which ultimately leads to an improvement in the network's overall performance.

The second improvement that was achieved was an improvement to the discriminator by employing the concept of relativistic average GAN (RaGAN). A relativistic discriminator, as opposed to a standard discriminator, attempts to forecast the probability that a real image is considerably more realistic than a fake image. Where else, Standard discriminators provide only the probability that an image is either real or fake.

Instead of using features of the VGG after activation, like in SRGAN, the perceptual loss is implemented by using features of the VGG before activation. When the features of the ESRGAN are used before activation, the perceptual loss in the ESRGAN is improved, which could lead to the recovery of brightness consistency and texture.

The SRGAN has a very limited number of activated features, which results in inadequate supervision and, as a consequence, poor performance. In addition, the active characteristics result in uneven brightness when compared to the original ground truth image.

However, approaches such as SRGAN and ESRGAN could be helpful in finding a solution to our dilemma. Despite this, these approaches are predicated on the concept of a bicubic downsampling kernel, which is distinct from real image degradation. For instance, sometimes results are focused on

In other words, assume a scenario where a CCTV has captured vehicle's image (video) but due its old-dated technology, it captures noise (unknown). In spite of, above mentioned methods can generate high resolution images, but they are unable to reconstruct image with region of these unknown noises (generated from and after the device). [12] Thus," the compensated high-frequency details such as image edges may cause inconsistency with the HR ground truth images".

Real-ESRGAN:

Mainly due to the fact that these traditional degradation models could, to some extent, generate synthetic training pairings. However, models that just include a simple degradation, which the study refers to as "first-order modelling," or classical degradation like above discussed methods, are unable to predict the complex degradations that occur in the real world since these degradations are typically comprised of a number of different procedures.

Therefore, Wang et al., n.d. [31]proposed finding a solution to the problem of modelling degradation and improving a number of different features of the synthesis network. High-order degradations are something they recommend in their essay as a way to more accurately imitate real-world degradations. They believed *sinc* filters can be used to combat the screeching and overrun problems caused by degradation. They suggested that the discriminator for Real-ESRGAN needs to have a bigger discriminative power for complicated training outputs than ESRGAN does since Real-ESRGAN intends to handle a considerably broader degradation space than ESRGAN does.

They implement a U-Net structure, which results in an improvement in comparison to the initial VGG-style discriminator. A more effective discriminator achieved through the utilisation of the U-Net architecture with spectral normalising regularisation (SN).

In addition to this, a number of potential enhancements to the synthesis network are proposed. The original ESRGAN's generator backbone has been retained in its entirety, including the residual-in-residual dense block (RRDB). However, in order to accept input images of varied resolutions, the input layer is modified so that the reversal of pixel shuffling, is used. This decreases the spatial size while simultaneously increasing the number of channels, which improves the system's overall efficiency. And spectral normalisation regularisation is utilised in order to achieve stable training. These alterations bring about surprise and significant improvements to the fabricated image.

Therefore, in order to generate data with good recognisable texture, I am using R-ESRGAN.

And due time limit on the project, I used latest pre-trained model from original GitHub repo.

In Results, I have applied this model on Dataset_1.And Code is in appendixes.

YOLOv4 for Car (object detection).

In this section, I will be discussing about different models used for detection. Discuss, best approach and execute it.

Field of detection techniques.

To improve the accuracy of object detection in the frame, after 2012, CNNs were developed in two directions. The first was optimization, which included batch normalisation, the Visual Geometry Group's (VGG) Google Neural Network (VGGNet), and Deep Residual Learning (ResNet). The improvement of the image processing framework was the alternative direction. The challenge of boundary selection was highlighted because there were multiple objects in the frame, despite CNN's excellent feature extraction and classification capabilities, which had good results when dealing with images containing a single object. These are two-staged methods (mostly regional recognising based).

R-CNN:

Proposed by Girshick et al., 2013 [Gi_1] In order to identify specific and classify objects, they presented a network that comprises of two stages. The first step involves applying high-capacity convolutional neural networks to bottom-up region proposals. The second is a paradigm for big CNN training with little labelled training data. They tested the PASCAL VOC dataset after fine-tuning and obtained a mAP of 58.6% And later in 2015 Fast R-CNN[8], In order to improve accuracy and reduce time taken by ConvNet, Girshick proposed improvement to R-CNN. And he run test applied with VGG16 on MSCOCO and got mAP of 66.9%.

Faster CNN:

Later in 2015, Girshick, Ren and team, proposed a Faster version of FRCN, by using conv for generating region proposal, and added two Conv layer. One converts each conv map position into a brief feature vector, while the other generates a probability score of the subject and regressed boundaries for 'k' region recommendations in relation to different sizes and aspect ratios at each conv map position. The methodology they designed enables a unified, deep-learning-based object detection system to perform at 5–17 frames per second. This method makes ROI proposals, each box proposal is used to extract feature in region map to identify which class this object belongs to. However, Region-based Fully Convolutional Network (R-FCN) by (Dai et al., n.d.)[24],added another convolutional layer namely Positive Scoremap,

which act as scoring map accordingly to each class detected in the frame .Both Faster RCN and R-FCN uses ROI proposal and method of extraction is also same but in R-FCN , it generates new proposal scoremap once only and ROI is used to vote scoremap, the idea was rather than using few conv nets this methodology uses to full extend on an entire image. Thus, it makes faster detection than F-RCN. Testing results showed mAP of 83.6% on PASCAL VOC2007 dataset and this method can detect object in 170ms which makes these methods 2.5-20 times faster than R-CNN family. But these methods are not applicable in real time scenario without extra computational power.

Here comes YOLO, which is state or art detector which works faster in the real time scenario without extra computational power.

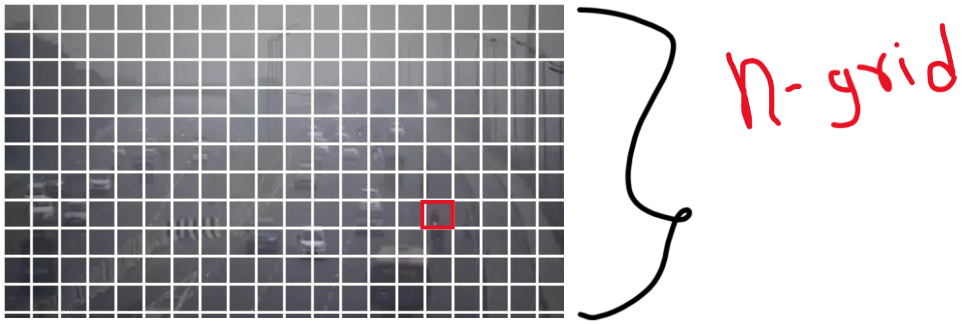
YOLO You Only Look Once:

As compared to more traditional techniques like the Deformable Part model [10], which employs a sliding window and run classification based on the highlighted shade (around or over the object). The detection rate is higher when the classification score is higher. As was already mentioned, region based CNNs perform selective search and classify certain regions using efficient classifiers around the object and detect object. Therefore, both utilize these local classifiers consistently to carry out detection network. The term "YOLO" (You Only Look Once) refers to a new method of object identification that Redmon et al [Redmon]. came up with new parameters for object detection and introduced in 2016. YOLO uses a single neural network to carry out object detection in a single step, hence the name "You Only Look Once". Additionally, as our problem statement requires for fast enough real-time vehicle detection, this approach outperformed earlier regional models like DM and R-CNN.

Principle:

In YOLO, imagine a single frame of over laying “n” number of grids with “s x s” dimension as shown in figure below, and just like in regional based networks, each cell in this grid is responsible for predicting different structures or shapes(in the image).For instance, red highlighted cell shown in figure, is responsible for predicting boundaries of the cell surrounding the object and assign a confidence value $P_{(c)}$, it is probability that cell contains an object. And vice-versa, lower the confidence value, lower the probability of object.

When visualized of these predictions together in each grid cell, a map of all the object in the image and boxes assigned highlighting with their confidence value. Now, for each object detected, there is a class probability, this is conditional probability for example, for red cell box in the shown figure that if it predicts cell contains a motorcycle, then there is a higher probability that “object” belongs to class “motorcycle”.



In other words, assume for any input image, each grid cell predicts B bounding boxes and each bounding box is consists of 5 prediction: x, y,w,h and confidence. Where “x” & “y” are coordinates of object with respect to centre of the most relevant to the object class or cell. And “w” & “h” are width and height with respect to object in the image.

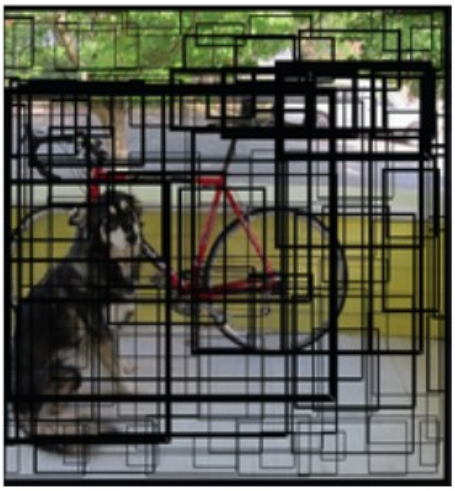
Intersection Over Union:

A common metric used to assess localization precision and determine localization faults in object identification models is intersection over union. The intersecting area between the bounding boxes for a certain prediction and the ground truth bounding boxes of the same area is the starting point for calculating the IoU with the predictions and the ground truth (Rezatofighi et al., 2019)[27]. The total area covered by the two bounding boxes, also known as the Union, is then calculated.

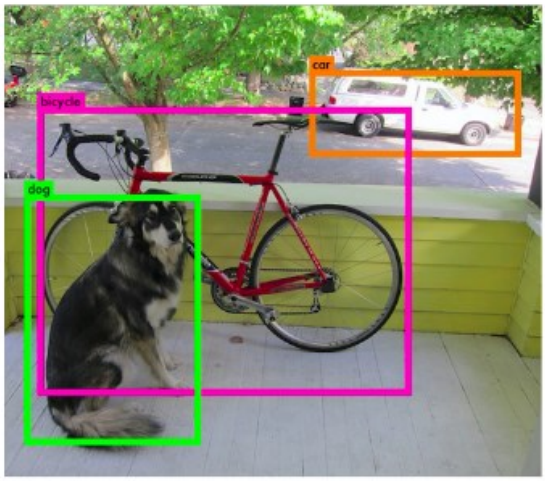
The ratio of the overlap to the total area, which is obtained by dividing the intersection by the union, offers us a decent indication of how well the bounding box resembles the original prediction. i.e., IOU = $\frac{Area\ of\ overlap}{Area\ of\ union}$

Non-Max Suppression:

As each box containing object pops up, as shown in below figure, it looks clustered like in left and ideally an object should be confine in a single box like on right image. This is possible due to Non-Max Suppression; his technique is used to “suppress” the less likely bounding boxes and keep only the best one.



Multiple Bounding Boxes



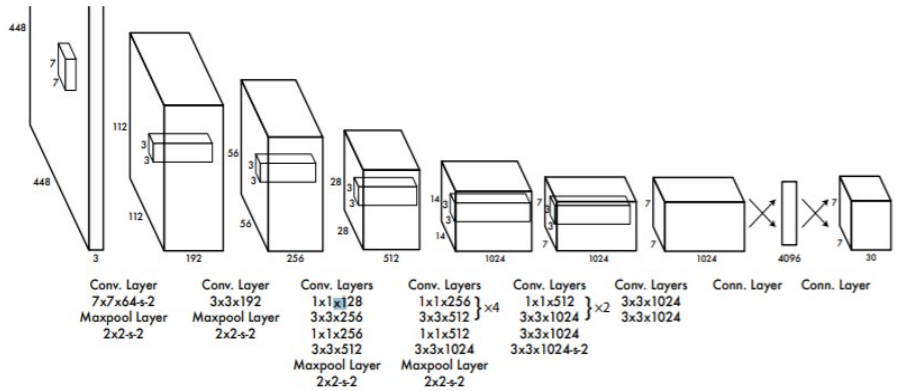
Final Bounding Boxes

Source: https://pjreddie.com/darknet/yolov1/

In our case, assume a vehicle with different number of confidence score boxes. The non-max suppression will first select the bounding box with the highest confidence score. And then remove all the other boxes with high overlap.

Architecture of YOLOv1:

Inspired from Googlenet architecture, this model has 24 convolutional layers and 2 fully connected layers as shown in below figure. Alternating 1×1 convolutional layers reduce the features space from preceding layers. Thus, in order to make prediction faster, reducing number of convolutional layers would be effective.



Source : https://arxiv.org/pdf/1506.02640.pdf

As model perform detections all together, so model incorporates global context (just like human forms understanding of object's attributes) so networks learn about the object's location, dimension, etc. which makes training process easy.

Training: The convolution layers are pretrained on ImageNet 1000-class dataset. Starting with each image with label is taken as “ground truth”. Assume a dataset of images with different vehicle, consider label boxes has confidence of “1”. These convolutional layers will be trained for for “n” numbers of iterations (term widely used for number of pipelines completed with respect to time) till confidence scores reaches nearly to “1” (which is highly dependent on volume of dataset).

Loss Function:

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

As shown in figure above, the loss function has following terms:

Coordinates: Grid cell offsets, whose values range from 0 to 1, are used to parameterize the x,y coordinates. Thus, the sum of squared error is employed. (First line of equation)

Dimensions: The width and height are proportional to the entire image. Therefore, squared error cannot be used because even a minor change might have a significant effect. By obtaining the square root and then determining the squared error, this is therefore partially resolved. (Second line of equation)

Confidence: Many grid cells in each image (the confidence) are empty of any objects. As a result, the model becomes unstable and the "confidence" scores of those cells are pushed towards 0, frequently overwhelming the gradient from cells that actually contain objects. As a result, noobj=0.5 reduces the loss from confidence predictions for boxes without objects. (Third and fourth line in the equation)

Class probabilities when there is an object, (fifth line of equation)

Evolution of YOLO:

YOLOv2(YOLO9000):

Although improvements in YOLOv1 compare old methods, there is a localization errors and YOLO has relatively low recall compared to region proposal-based methods (like F-RCN,R-FCN).So in order to overcome these barrier, [Redmon] came with new upgrades.Like 1) Batch Normalization for more stable training of deep neural networks by stabilizing the distribution of the input layers during training.The

goal of this approach is to normalize the features (the output of each layer after passing activation) to a zero-mean state with a standard deviation of 1 By applying batch normalization followed by all YOLOv2's convolution layers. This technique not only reduces training time but also increases the generalization of the network. In YOLOv2, batch normalization increased mAP (mean average precision) by about 2% (Redmon, et al., 2016).2) High Resolution Classifier, help the model “adapt” to a large resolution of 448 × 448 instead of suddenly increasing the image size when the feature extractor training phase moves to the object detector training phase.3) Convolutional with the anchor box, so that model can detect more object in a single bounding box(if any).

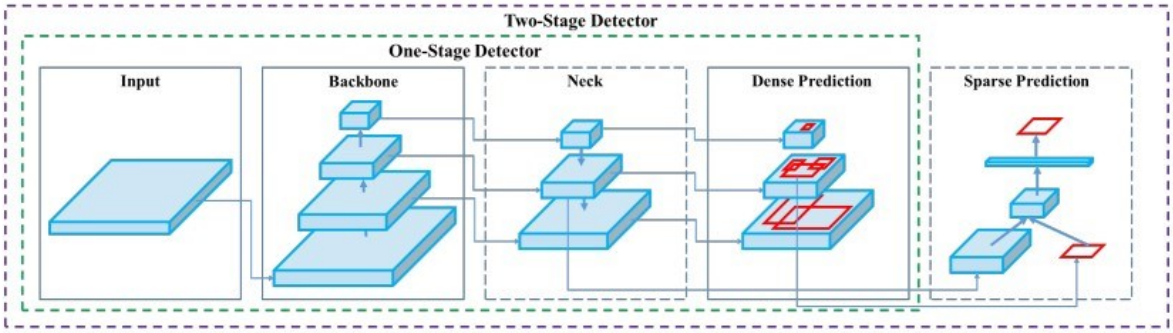
Even though this improvement, in starting stage of this project, I conducted detection on dataset, it mAP of 51%.

YOLOv3:

In contrast to YOLOv1's 11 layers, YOLOv2's customized deep 30-convolutional layers are used in the Darknet architecture. More layers for deep neural networks correlate with improved efficiency. However, when moving to deeper layers, the image quality was down sampled, which resulted in the loss of fine-grained details. Thus this will brings us back to problem of low resolution classification. Better architecture was developed by YOLOv3, which combined elements of YOLOv2, Darknet-53 (53 convolutional layers), and Residual networks (ResNet) (Redmon, et al., 2018). The bottleneck structure (1x1 followed by 3x3 convolution layers) is used to construct the network, along with a skip connection, inside each residual block. The Darknet-53 design, which originally included a 53-layer network, was employed by the model to train a feature extractor. Following that, 53 further layers were added to the stack for the detection head of the training object detector, giving YOLOv3 a total of 106 layers in the underlying convolutional architecture.

YOLOv4

Any object detector whether it is a single stage detector or two-staged detector, has a common architecture just like in below figure, where Dense Prediction and Sparse Prediction are head for One-stage and Two-stage respectively.



Source: https://arxiv.org/pdf/2004.10934v1.pdf

In 2020, Bochkovskiy, Wang and Liao, n.d. [2] proposed an advancement in YOLO series, YOLOv4.With the help of “selection criteria”, idea of mix-matching with basic blocks like Backbone, Neck/s and Head (as YOLO is single stage)(And not whole architecture).

Backbone:

In a network, many convolutional layers are connected but rather than using output from last layer, it takes outputs from all previous layers as well as input origin. These are dense blocks in YOLOv4. Interconnected many numbers of these dense blocks makes Dense net and it is connected to a transition layer. Where else, The CSP (Cross Stage Partial) is built on the same theory as the aforementioned DenseNet, with the exception that the input will be divided into 2 halves rather than using the foundation layer's full-size input feature map. A chunk will proceed normally through the dense block, while another will be delivered without being processed directly to the next stage. Therefore, combining Darknet53 from YOLOv3 and CSP, author proposed CSPDarknet53. CSP optimizes the network to retain features, lowers the amount of network parameters, and maintains fine-grained features to helps separating to greater depths more effectively.

Neck/s:

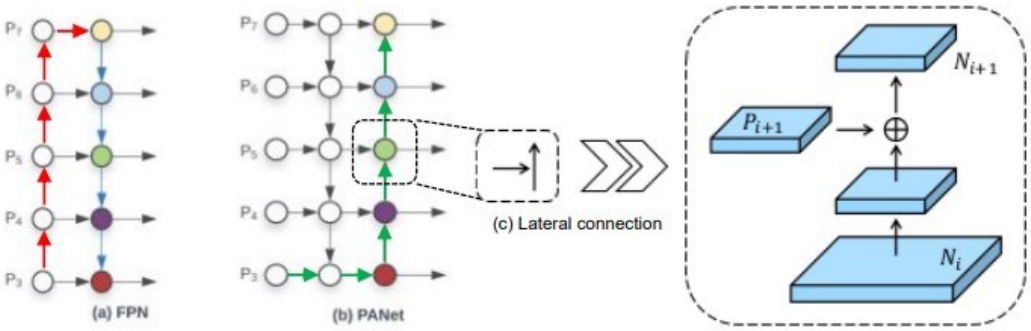
The purpose is to add extra layers between the backbone and head (prediction block).

SPP Blocks:

In YOLOv4, a modified version of SPP (Spatial Pyramid Pooling) layer is used to retrained output dimension. A 1 x 1 convolution is employed between the backbone and the SPP block to reduce the amount of input feature maps delivered to the SPP block (from 1024 to 512). Following that, input feature maps are duplicated and pooled according to the same principle as a traditional SPP block, with the exception that padding is used to maintain a constant size for the output feature maps. Three feature maps will then maintain their original sizes of size*f*map x size*f*map x 512. Thus, making classification smooth.

PANet (Path Aggregation Network):

Unlike FPN (Feature Pyramid Networks) which localize spatial feature travels upward to the top, usually it requires 100 layers(more in many cases), in order to minimize this , in PAN its takes 10 layers which make fine grain features available. With layers that yield feature maps with the same spatial sizes at each stage, the bottom-up augmentation path can be seen as a same as of the FPN top-down path. By using an element-wise addition operation, these feature maps are coupled to the lateral architecture, as opposed to the modified PAN architecture for YOLOv4.



(a) FPN architecture. (b) PAN architecture. (c) Connection in bottom-up augmentation path. (Solawetz, 2020)

Credit: https://jonathan-hui.medium.com/yolov4-c9901eaa8e61

Thus, giving better performance to detection(detector)

Head:

It is responsible for performing detection, in our case head is YOLO. As mentioned above, the final prediction is made up of a vector that includes the probability classes, the prediction confidence score, and the predicted bounding box coordinates (centre, height, and width).

Due to its slow detection speed, the region-based Fast Convolution Network, which has great detection accuracy, is unable to meet the real-time need for surrounding target identification when the vehicle is in motion. Where else, YOLOv3 (a version of YOLO) with incredibly high detection speeds but poor detection accuracy. Therefore, I choose YOLOv4 for detection part.

As, YOLOv4 provide following features.

- As discussed earlier Higher resolution have more contextual information which makes detection process more accurate, but it comes at cost of high computational power, as it requires more convolutional and output layers, YOLOv4 provides with optimal balance with selection criteria.(“Bag of Freebies” and “Bag of specials” [Original Paper])
- As Darknet53 provides with one of the best parameter aggregations in (or when) less receptive field. (Making prediction faster)
- Model provides with data augmentation like mosaic, blur, MixUp, etc., widening the limited datasets and at same time making training strategy better.
- Customize training for better classification and labelling class/es.
- Plot accuracies while training.

Therefore, due to perks like “Bag of freebies” , “ Bag of specials”, faster and accuracy I intent to train YOLOv4 and derive results to support my proposed hypothesis.

Training costumed YOLOv4:

Tool: Google Colab. (because of inbuild cuda and GPU)

Using the Dataset_yolov4 as the training set for Yolov4, a subset of photos containing the 4 classifications of automobile, ambulance, truck, and bus were retrieved (which are most commonly used vehicles on roads). Using [Roboflow], annotated labels were added manually, all pictures were enhanced same pixels resolution (resized) to 614*614, so that model would train with higher contextual information, and code for Roboflow environment was generated for Google Colab import.

Training and Validation:

Cloned the GitHub repository of the original author [AlexeyAB and Darknet] in order to train our custom dataset for training model to recognise these 4 classes (Car, Truck, Bus, and Ambulance). So, that it uses less layers and perform faster.

And other reason was to eliminate objects, as they would have less confidence score and it would look less clustered (not for project but in real time scenario.)

Training set: 778, Validation set: 222 and Testing set: 110

Additionally, a customised configuration for YOLOv4 was created by creating a "Config" file in the manner of P. J. Reddie, which defined the structure of the network based on Dataset yolov4. Darknet identifies networks layers since it iterates rather than "epochs", this dataset was trained for "8000" iterations.

And I was saving weights as well best weights at every 1000 iterations.

As I was facing few barriers, I trained YOLOv4 for 12 times and each time it took at least 11 hours for training. The graph below is from my 12^{th}-training attempt, with the “*help of freebies*” (from YOLOv4) accuracies of model is plotted at every 1000 iterations.

I executed training in following steps.

1. Connected to GPU.
2. Cloned and installed YOLOv4 repository.
3. Labelled Data using Roboflow.
4. Wrote YAML file containing information of dataset and paths.
5. Wrote config files for Training and Test
   With batch size: 64. (batch of numbers of images forwarded into network in one iteration. Thus, it would take total batch of 64) and Subdivision of 24. So that, at every iteration model will be trained.
6. Plotted graph for model accuracies at every 1000 iterations.
7. Trained for 8000 iterations.

For one iteration it took 4.98 s and total execution time was 11.59 hours.

Code is in the Appendix section.

Testing costumed YOLOv4:

Tool: Google Colab.

After training the model, and ready to verify my proposed hypothesis.

I executed testing in followings steps,

1. Connecting to GPU.
2. Cloning repository into colab environment.
3. To accelerate efficiency and building, I have use CUDA and CUDA GPU-enabled OpenCV and making Darknet environment.
4. Then downloaded our trained weights into the colab.
5. Then defining functions for plotting boxes and boundaries.

Result and Evaluation:

In this section, I will discuss the experimental outcomes that were obtained through the application of my methodology to datasets 1.

Results from R-ERGAN

I downloaded a pre-trained model to upscale images in order to answer our first research question and prove my hypotheses.

The following are the steps that I took in order to apply RESRGAN to dataset_1. I began by downloading the original author's GitHub repository and configuring the environment.

2. After that, download a model that has already been pretrained and set the directory to Dataset 1.

3. And last but not least, the upscaling of images with noise reduction.

Tool: Jupyter notebook (while learning) and Colab (while executing)

Processor: AMD RYZEN 5.

As a result of the above, R-ESRGAN was able to produce images that had good textures (up to a certain depth), which enabled us to validate our most recent research question and provide support for the hypothesis that detections will be smooth and accurate. Despite the fact sometimes it can produce unsightly textures at times (like smudge or strokes)

Example. Image of the right was 8x8 and image from left is 128x128. And I have considered(think) this as example of crop image from CCTV.

 from 

Therefore, this answers our first research question.

Code in the Appendix section.

Results without R-ESRGAN

As was covered in the section on methodology, trained weights can be used to determine the locations of vehicles in any given image. A bounding box will be highlighted around the vehicle and display the probability of that vehicle being present in the frame if and only if the presence of any vehicle is detected in the image.

After the model is trained on the training set, its performance on validation and test data is evaluated. Deep learning models can be evaluated using different metrics, depending on the problem statement and application.

Precision, Recall, Confusion-matrix, PR-curve, etc. are common image classification metrics. mIoU is used for picture segmentation. mAP is an object detection assessment measure that considers both item category(class) and location. (Basically, its metric for accuracy of model while performing detection)

In order to answer my second research question, before doing the upscaling, I ran the YOLOv4 detection algorithm on dataset_1, and the results are as follows.

IOU=0.5:0.95

Class	Detections by YOLOv4.	Detection by me.
Ambulance	6	9
Truck	2	6
car	34	66
bus	1	4

mAP = 37.93 %

While testing on generated dataset. I manually verified the accuracy. And I got following results.

Where, Ground Truth is number of classes I observed.

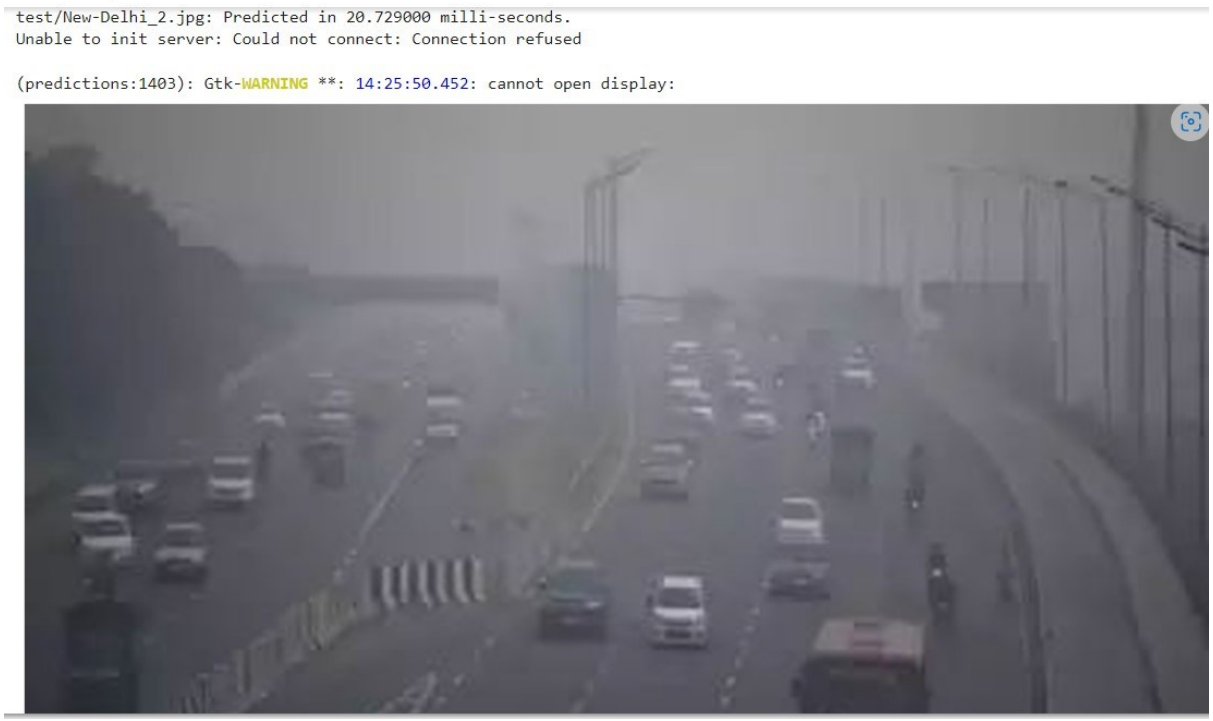
IoU threshold = 0.5:0.95

Class	True positive	False positive	Ground truth	AP (average precison)
Ambulance	8	1	9	88.88%
Truck	4	0	4	100%
Car	34	13	34	72.34%
bus	7	1	26	87.5%

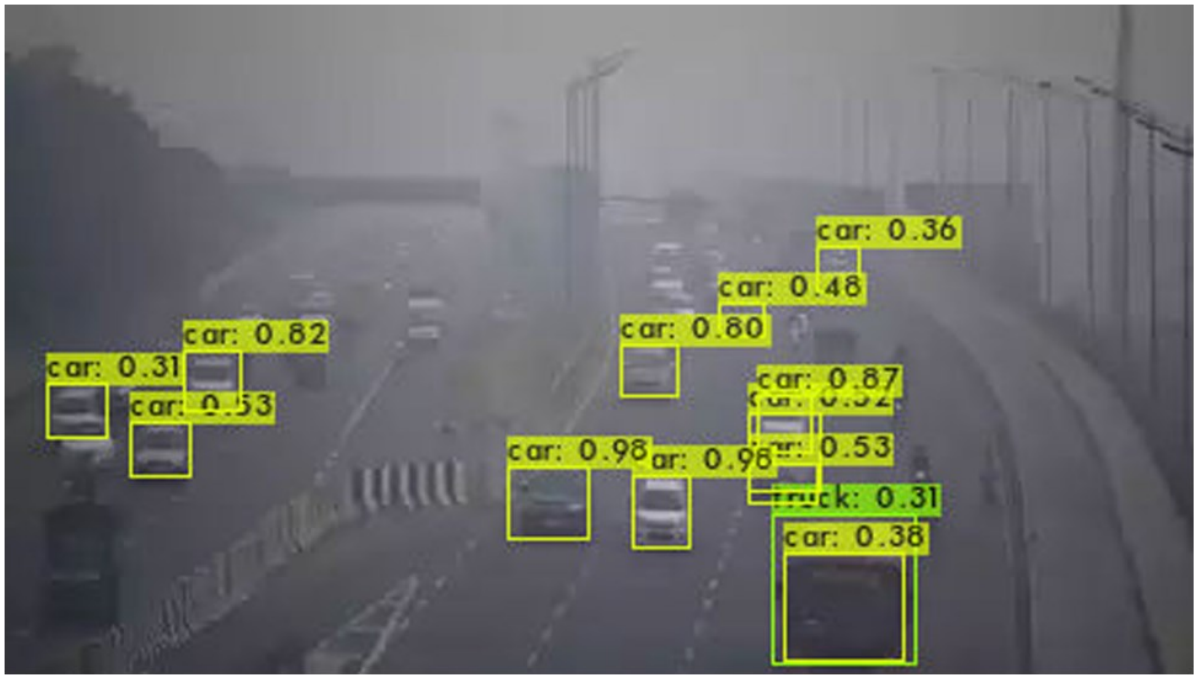
mAP = 83.01 %

Results from YOLOv4

#Image before applying R-ESRGAN:



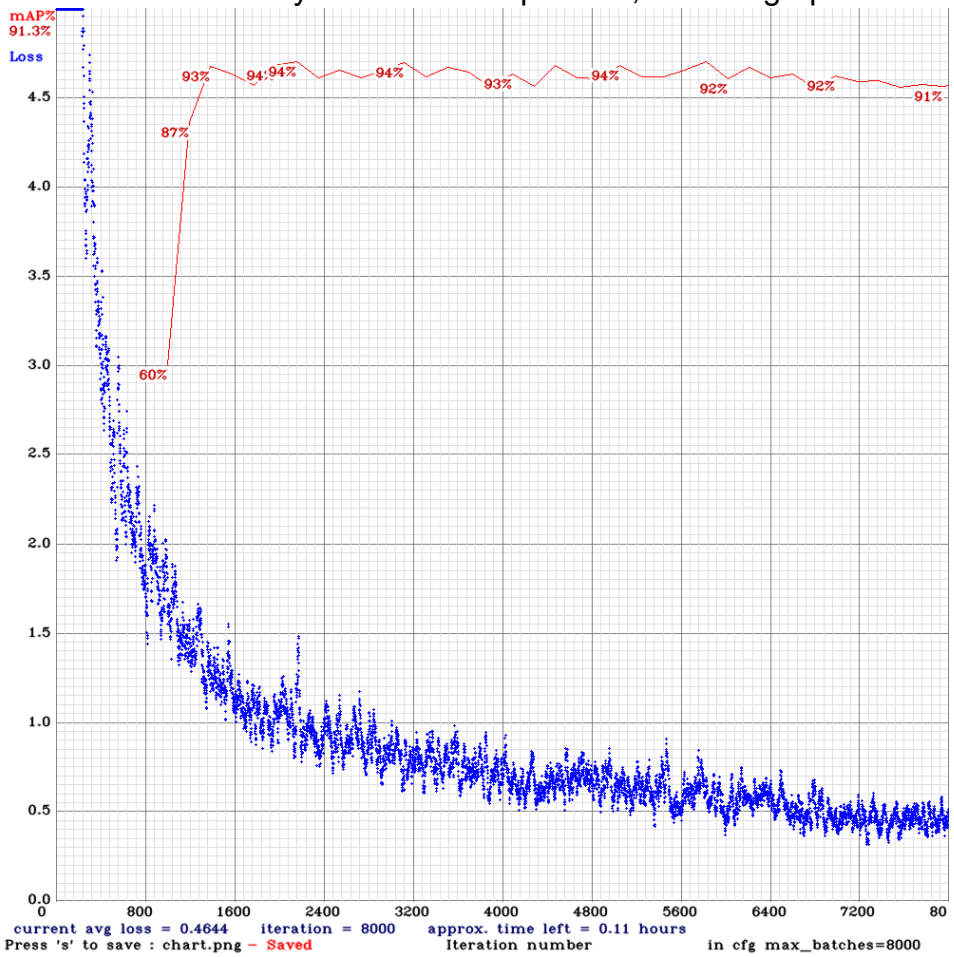
#Images after applying R-ESRGAN:



Therefore, this answers our second research question, that R-ESRGAN assists YOLOv4 in detecting vehicle from low resolution data.

Training results

In order to answer my last research question, I derive graph for training.



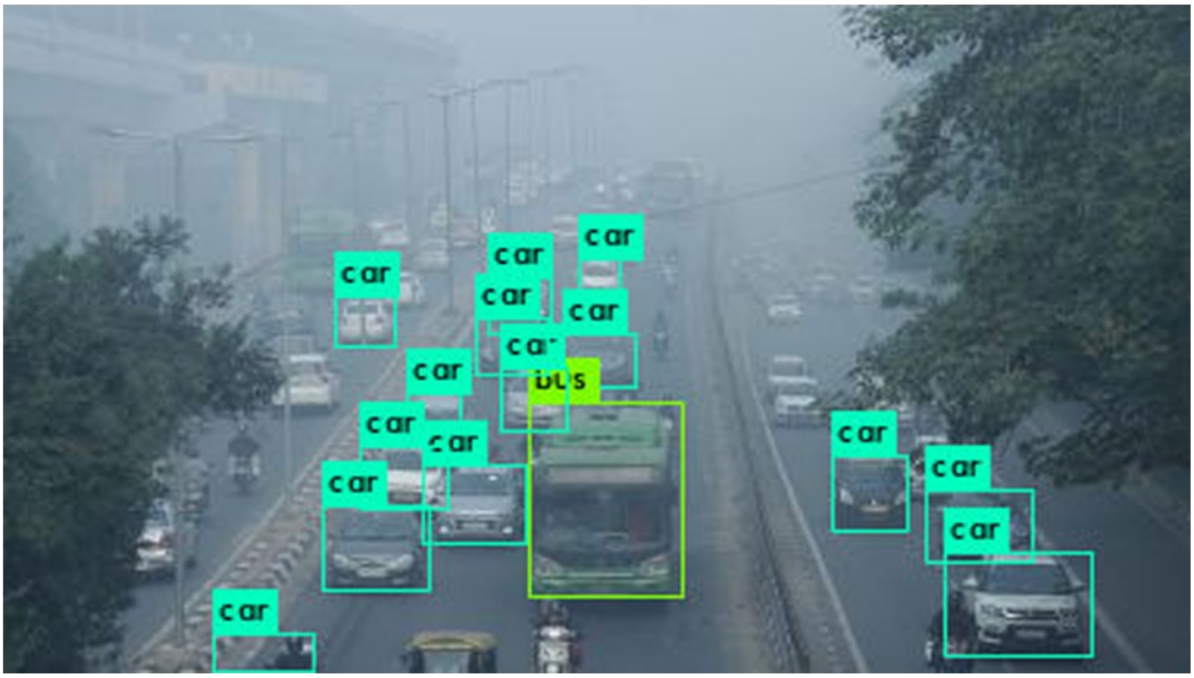
In above graph Loss vs Class detections w.r.t Number of iteration is plotted, as graph depicts losses were decreasing gradually at every (almost) 141 of iterations after 1000 iterations.

Therefore, answer to my last research question is 94.87%

Test result.

In order to prove robustness of my trained YOLOv4 model, I executed detections.

Detection on a low pixel’s image:



Detection on a high pixels image:

Although my proposed hypothesis is true that R-ESRGAN assists with upscaling data to make quick and accurate predictions from YOLOv4, still out of 12 images my proposed worked on only 8 images.

Conclusion and Future work:

Over the past few decades, global vehicle and population growth have increased traffic problems. Many smart cities or urban areas around the world have CCTV cameras installed to ensure netizens' safety and convenience. It includes traffic management. Vehicle detection is crucial to traffic management. In the past 5 years, vehicle detection algorithms have become more accurate and faster. Deep learning techniques use neural network algorithms to mimic the human brain's ability to see patterns, link knowledge, and identify objects. But due outdated technology, Time and memory needed for traffic management recognition, sometimes enlarging the frame (image) to detect vehicles can degrade pixel density, affecting detection.

Therefore, started with research questions my guide, I proposed generating high-resolution images from low-resolution pictures with bare minimum noise using R-ESRGAN and then identifying cars/vehicles from the generated images using my trained YOLOv4 and executed it with satisfactory results.

In the process, I trained YOLOv4 using Dataset_yolov4. and got accuracy of almost 95 %. On the other hand, I tested my proposed methodology on Dataset_1. In spite of It was concluded that, this methodology works with both low-resolution images as well as high resolution images, I strongly believe that rather using pre-trained GAN model, results would have been more satisfactory with trained GAN model.

In this paper, I have worked on couple of open-source datasets for both learning as student and for execution. Due to time restrain and less available datasets in public domain, I was not able to prove my hypothesis but in future, I would like make dataset of 5K images with different types of noises and execute with proposed methodology. And as an individual work in the future, I am planning to train R-ESRGAN with new dataset for reconstructing and upscaling images which would give me insights I lacked in this project.

References:

[1] Bautista, C.M., Dy, C.A., Mañalac, M.I., Orbe, R.A. and Cordel, M., 2016. Convolutional neural network for vehicle detection in low resolution traffic videos. [online] Convolutional neural network for vehicle detection in low resolution traffic videos. Available at: <https://ieeexplore.ieee.org/document/7519418> [Accessed 23 Jun. 2022]

[2] Bochkovskiy, A., Wang, C.-Y. and Liao, H.-Y. (n.d.). *YOLOv4: Optimal Speed and Accuracy of Object Detection*. [online] Available at: https://arxiv.org/pdf/2004.10934v1.pdf.

[3] [Ca] Cao, L., Wang, C. and Li, J. (2016). Vehicle detection from highway satellite images via transfer learning. *Information Sciences*, 366, pp.177–187. doi:10.1016/j.ins.2016.01.004.

[4] Chen, Y., Qin, R., Zhang, G. and Albanwan, H. (2021). Spatial Temporal Analysis of Traffic Patterns during the COVID-19 Epidemic by Vehicle Detection Using Planet Remote-Sensing Satellite Images. *Remote Sensing*, 13(2), p.208. doi:10.3390/rs13020208.

[5]Choudhury, A. (2020). *Top 8 Algorithms For Object Detection One Must Know*. [online] Analytics India Magazine. Available at: https://analyticsindiamag.com/top-8-algorithms-for-object-detection/.

[6] Dai, J., Li, Y., He, K. and Sun, J. (n.d.). *R-FCN: Object Detection via Region-based Fully Convolutional Networks*. [online] Available at: https://arxiv.org/pdf/1605.06409.pdf [Accessed 9 Jul. 2022].

[7] Dong, C., Loy, C., He, K. and Tang, X. (n.d.). *Image Super-Resolution Using Deep Convolutional Networks*. [online] Available at: https://arxiv.org/pdf/1501.00092.pdf.

[8] Girshick, R. (2015). *Fast R-CNN*. [online] arXiv.org. Available at: https://arxiv.org/abs/1504.08083.

[9] Girshick, R., Donahue, J., Darrell, T. and Malik, J. (2013). Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv:1311.2524 [cs]*. [online] Available at: http://arxiv.org/abs/1311.2524v1 [Accessed 9 Jul. 2022].

[10] Girshick, R. (n.d.). *Deformable part models*. [online] Available at: http://vision.stanford.edu/teaching/cs231b_spring1213/slides/dpm-slides-ross-girshick.pdf [Accessed 3 Jul. 2022].

[11] Ji, H., Gao, Z., Mei, T. and Ramesh, B. (2019). Vehicle Detection in Remote Sensing Images Leveraging on Simultaneous Super-Resolution. *IEEE Geoscience and Remote Sensing Letters*, [online] Volume: 17, Issue: 4, April 2020)(4), pp.676–680. doi:10.1109/LGRS.2019.2930308.

[12] Jiang, J., Ma, J., Wang, Z., Chen, C. and Liu, X. (2019). Hyperspectral Image Classification in the Presence of Noisy Labels. *IEEE Transactions on Geoscience and Remote Sensing*, 57(2), pp.851–865. doi:10.1109/tgrs.2018.2861992.

[13] Johnson, J., Alahi, A. and Fei-Fei, L. (n.d.). *Perceptual Losses for Real-Time Style Transfer and Super-Resolution*. [online] Available at: https://arxiv.org/pdf/1603.08155.pdf.

[14] Khandelwal, Y. (2021). *Image Super Resolution | Deep Learning for Image Super Resolution*. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2021/05/deep-learning-for-image-super-resolution/ [Accessed 14 Jul. 2022].

[15] Kim, J., Lee, J., Song, K. and Kim, Y.-S. (2019). Vehicle model recognition using SRGAN for low-resolution vehicle images. *Proceedings of the 2nd International*

Conference on Artificial Intelligence and Pattern Recognition - AIPR ’19. doi:10.1145/3357254.3357284.

[16] Krizhevsky, A., Sutskever, I. and Hinton, G.E. (2012). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6), pp.84–90. doi:10.1145/3065386.

[17] Ledig, C., Theis, L., Huszar, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z. and Shi, W. (2016). *Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network*. [online] arXiv.org. Available at: https://arxiv.org/abs/1609.04802.

[18] Mahendru, M. and Dubey, S.K. (2021). *Real Time Object Detection with Audio Feedback using Yolo vs. Yolo_v3*. [online] IEEE Xplore. doi:10.1109/Confluence51648.2021.9377064.

[19] Maity, M., Banerjee, S. and Sinha Chaudhuri, S. (2021). *Faster R-CNN and YOLO based Vehicle detection: A Survey*. [online] IEEE Xplore. doi:10.1109/ICCMC51019.2021.9418274.

[20] Nadipally, M. (2019). Optimization of Methods for Image-Texture Segmentation Using Ant Colony Optimization. *Intelligent Data Analysis for Biomedical Applications*, pp.21–47. doi:10.1016/b978-0-12-815553-0.00002-1.

[21] Pang, J., Li, C., Shi, J., Xu, Z. and Feng, H. (2019). $\mathcal{R}^2$ -CNN: Fast Tiny Object Detection in Large-Scale Remote Sensing Images. *IEEE Transactions on Geoscience and Remote Sensing*, 57(8), pp.5512–5524. doi:10.1109/tgrs.2019.2899955.

[22] Peng, H., Wang, X., Wang, H. and Yang, W. (2014). Recognition of Low-Resolution Logos in Vehicle Images Based on Statistical Random Sparse Distribution. *IEEE Transactions on Intelligent Transportation Systems*, pp.1–11. doi:10.1109/tits.2014.2336675.

[23] Rabbi, J., Ray, N., Schubert, M., Chowdhury, S. and Chao, D. (2020). Small-Object Detection in Remote Sensing Images with End-to-End Edge-Enhanced GAN

and Object Detector Network. *Remote Sensing*, 12(9), p.1432. doi:10.3390/rs12091432.

[24] Ren, S., He, K., Girshick, R. and Sun, J. (2015). *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. [online] arXiv.org. Available at: https://arxiv.org/abs/1506.01497.

[25] Redmon, J., Divvala, S., Girshick, R. and Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [online] doi:10.1109/cvpr.2016.91.

[26] Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. arXiv. Seach date 20.11.2020. https://arxiv.org/pdf/1804.02767.pdf

[27] Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I. and Savarese, S. (2019). Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression. *arXiv:1902.09630 [cs]*. [online] Available at: https://arxiv.org/abs/1902.09630.

[28] Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A., Bishop, R., Rueckert, D., Wang, Z. and Twitter (n.d.). *Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network*. [online] Available at: https://arxiv.org/pdf/1609.05158.pdf.

[29] Tas, S., Sari, O., Dalveren, Y., Pazar, S., Kara, A. and Derawi, M. (2022). Deep Learning-Based Vehicle Classification for Low Quality Images. *Sensors*, 22(13), p.4740. doi:10.3390/s22134740.

[30] Wang, X., Yu, K., Wu, S., Gu, J., Liu, Y., Dong, C., Loy, C., Qiao, Y. and Tang, X. (n.d.). *ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks*. [online] Available at: https://arxiv.org/pdf/1809.00219.pdf [Accessed 16 Jul. 2022].

[31] Wang, X., Xie, L., Dong, C. and Shan, Y. (n.d.). *Real-ESRGAN: Training Real-World Blind Super-Resolution with Pure Synthetic Data*. [online] Available at: https://arxiv.org/pdf/2107.10833.pdf [Accessed 16 Jul. 2022].

Reference for Dataset:

[**SANDEEP**]. [2020/June/16]. [Vehicle Data Set], [CC0: Public Domain].

Link: Dataset_yolov4.

[1764 words to exclude]

Appendix:

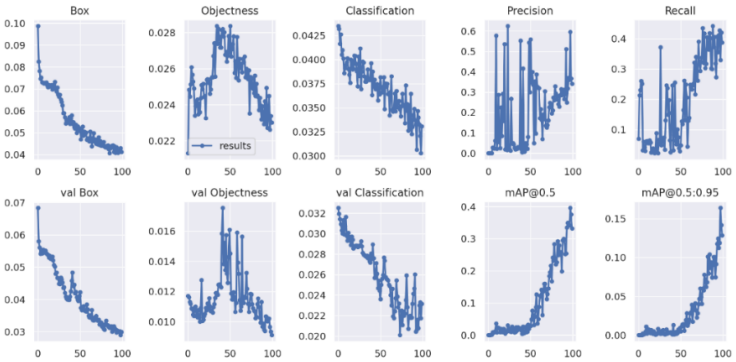
Challenging barriers faced and overcome:

- I was facing problem while training and model was making error while distinguishing between Truck, Bus and Ambulance. (Due to low number of images)
- Due to mistakes in labelling at early stage, trained model used to detect just one object per detection.
- As I was using Google colab for YOLOv4 model, every time runtime was getting disconnected after a while. Thus, it took me 12 attempt to save weights and every file as backup to drive.

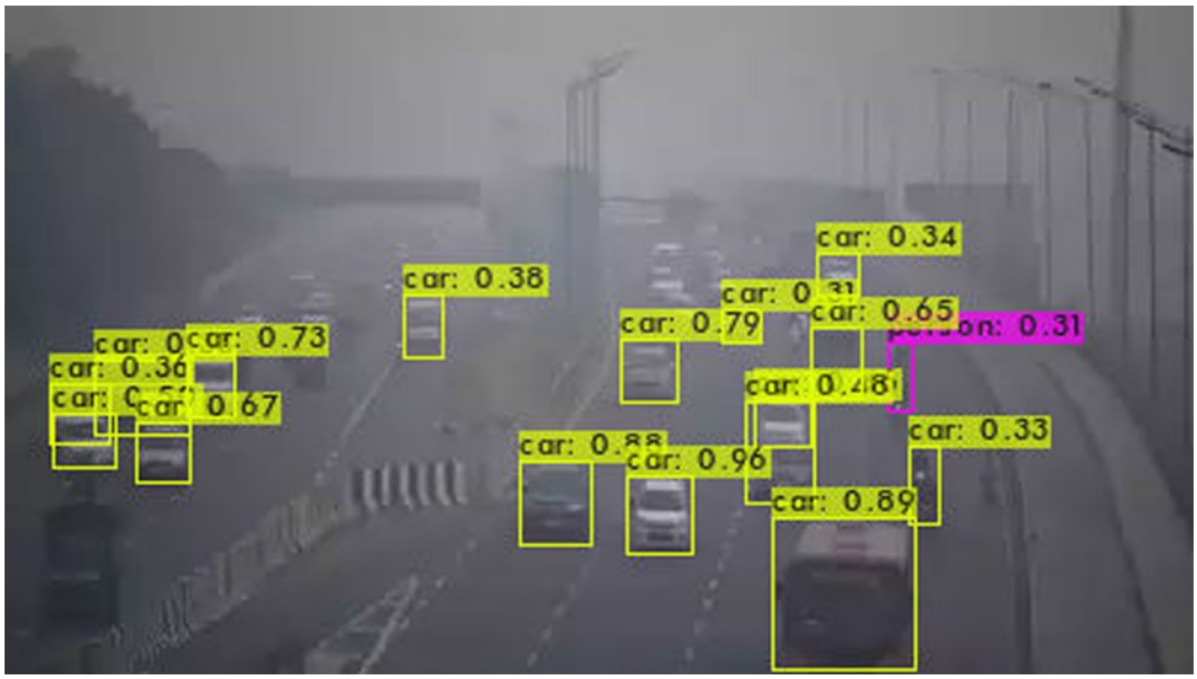
Challenge yet to overcome:

- Due latest pre-trained R-ESRGAN model is trained for reconstructing human face, therefore it is very difficult to reconstruct objects (multiple objects at a time). Due to which accuracy of proposed methodology is versatile.

Below figure is from one of my trainings, where model accuracy was up to 97.6% but unfortunately due to error in the labelling while training, this model was unable to detect more than one vehicle.



And here is collage of detections (note: Even though this could not detect more than one vehicle in the frame, it worked!




```
 Detection layer: 161 - type = 28
/content/New-Delhi_2.jpg: Predicted in 54.607000 milli-seconds.
car: 36%
car: 50%
car: 33%
car: 67%
car: 73%
car: 38%
car: 88%
car: 79%
car: 96%
car: 31%
car: 48%
car: 46%
car: 89%
car: 65%
car: 34%
person: 31%
car: 33%
Unable to init server: Could not connect: Connection refused

(predictions:2162): Gtk-WARNING **: 22:30:04.558: cannot open display:
```

Sometimes due to latest pre-trained R-ESRGAN model, and coco.name encoded to YOLO, some results were unexpected as shown in above figure.

Code for generating images using R-ESRGAN.

#Cloning Real-ESRGAN from original GitHub repo.

Downloading Pre-trained model.

```
!wget https://github.com/xinntao/Real-ESRGAN/releases/download/v0.1.0/RealESRGAN_x4plus.pth -P experiments/pretrained_models
```

---

installing nessary libraries.

```
import os.path as osp
import glob
import cv2
import numpy as np
import torch
import RRDBNet_arch as arch
```

Setting path to downloaded pretrained model

```
path_m = 'content/RRDB_ESRGAN_x4.pth'
```

---

setting torch device to cuda.

```
device = torch.device('cuda') #CPU
```

Setting Dataset_1 as input directory

```
Input = 'Dataset_1/*'
```

Generating images : source https://github.com/xinntao/Real-ESRGAN

[ ]

```
model = arch.RRDBNet(3, 3, 64, 23, gc=32)

model.load_state_dict(torch.load(path_m), strict=True)

model.eval()

model = model.to(device)

idx = 0

for path in glob.glob(input):
```

```
    idx += 1

    base = osp.splitext(osp.basename(path))[0]

    print(idx, base)
```

---

read images Source : https://github.com/xinntao/Real-ESRGAN

[]

```
img = cv2.imread(path, cv2.IMREAD_COLOR)

    img = img * 1.0 / 255

    img = torch.from_numpy(np.transpose(img[:, :, [2, 1, 0]], (2, 0, 1)
)).float()

    img_LR = img.unsqueeze(0)

    img_LR = img_LR.to(device)

    with torch.no_grad():

        output = model(img_LR).data.squeeze().float().cpu().clamp_(0, 1
).numpy()

    output = np.transpose(output[[2, 1, 0], :, :], (1, 2, 0))

    output = (output * 255.0).round()

    cv2.imwrite('output/{:s}_new.jpg'.format(base), output)
```

Code For Training YOLOv4:

#Installing Darknet on Colab

clone darknet repo

!git clone https://github.com/AlexeyAB/darknet
%cd /content/darknet/
%rm Makefile

---

makefile to have GPU and OPENCV enabled

Source:https://universe.roboflow.com/

```
%cd darknet/
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile
!sed -i 's/GPU=0/GPU=1/' Makefile
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile
!sed -i 's/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile
!sed -i 's/LIBSO=0/LIBSO=1/' Makefile
```

---

built darknet

```
!make
```

Downloading latest weights

```
!wget https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.conv.137
```

Downloading labelled data from Roboflow.com

```
#Downloading Labelled dataset from Roboflow.com in Darknet Format
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="SkxnpLYogjwvMFaqTLW7")
project = rf.workspace("project-aagka").project("multi_object")
dataset = project.version(3).download("darknet")
```

Set up training file directories for custom dataset : source https://universe.roboflow.com/

```
%cd /content/darknet/
%cp {dataset.location}/train/_darknet.labels data/obj.names
%mkdir data/obj
#copy image and labels
%cp {dataset.location}/train/*.jpg data/obj/
%cp {dataset.location}/valid/*.jpg data/obj/

%cp {dataset.location}/train/*.txt data/obj/
%cp {dataset.location}/valid/*.txt data/obj/

with open('data/obj.data', 'w') as out:
  out.write('classes = 3\n')
  out.write('train = data/train.txt\n')
```

```
 out.write('valid = data/valid.txt\n')
 out.write('names = data/obj.names\n')
 out.write('backup = backup/')
```

---

write training file.

```
import os

with open('data/train.txt', 'w') as out:
 for img in [f for f in os.listdir(dataset.location + '/train') if f.endswith('jpg')]:
  out.write('data/obj/' + img + '\n')
```

Defining config file for 4 classes

```
#With objects list
#yolo.obj
#Source: https://github.com/AlexeyAB/darknet
def file_len(fname):
 with open(fname) as f:
  for i, l in enumerate(f):
   pass
 return i + 1

num_classes = file_len(dataset.location + '/train/_darknet.labels')
```

Writing config file for detector:

source: https://github.com/AlexeyAB/darknet

```
with open('./cfg/custom-yolov4-detector.cfg', 'a') as f:
 f.write('[net]' + '\n')
 f.write('batch=64' + '\n')
  f.write('subdivisions=24' + '\n')

 f.write('width=416' + '\n')
 f.write('height=416' + '\n')
 f.write('channels=3' + '\n')
 f.write('momentum=0.949' + '\n')
 f.write('decay=0.0005' + '\n')
 f.write('angle=0' + '\n')
 f.write('saturation = 1.5' + '\n')
 f.write('exposure = 1.5' + '\n')
 f.write('hue = .1' + '\n')
 f.write('\n')
```

```
 f.write('learning_rate=0.001' + '\n')
 f.write('burn_in=1000' + '\n')
```

Defining maximum numbers of iterations.

```
#Defining maximum number of batch
#Source https://github.com/AlexeyAB/darknet
 max_batches = 10000
 f.write('max_batches=' + str(max_batches) + '\n')
 f.write('policy=steps' + '\n')
 steps1 = .8 * max_batches
 steps2 = .9 * max_batches
 f.write('steps='+str(steps1)+','+str(steps2) + '\n')
```

Training YOLOv4

```
!./darknet detector train data/obj.data cfg/custom-yolov4-detector.cfg yolov4.conv.137 -
dont_show -map
```

---

downloading files as zip

```
# Backup
!zip -r /content/darknet_1.zip /content/darknet
files.download('/content/darknet_1.zip')
```

Code for detecting vehicles from the images using YOLOv4:

YOLOv4 uses C/CUDA neural network framework Darknet. Cloning the Darknet YOLOv4 Github repository.

```
!git clone https://github.com/AlexeyAB/darknet
```

To accelerate efficiency and building, I have use CUDA and CUDA GPU-enabled OpenCV.

```
 %cd darknet
 !sed -i 's/OPENCV=0/OPENCV=1/' Makefile
 !sed -i 's/GPU=0/GPU=1/' Makefile
 !sed -i 's/CUDNN=0/CUDNN=1/' Makefile
 !sed -i 's/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile
/content/darknet
```

Make darknet executable by running the make command.

```
!make
```

Downloading weights.

```
#download from drive
!gdown --id 1DiEMexKHpBrSmaQrWfE-o7uNqLlgpffE
!gdown --id 1YLJbPyo4WzcmXg3EWWENb3y_l26M0lMe
#!gdown --id 1-QLUvvYFSCBmHIbwPQiMNRC6Q_GuLFDR
!wget https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.weights
```

Now we develop the matplotlib function imshow() to visualise our predictions. Colab cannot output photos in new windows, therefore this is necessary. Matplotlib is required.

```
import cv2
import matplotlib.pyplot as plt
%matplotlib inline
def imShow(path):
  image = cv2.imread(path)
  height, width = image.shape[:2]
  resized_image = cv2.resize(image,(3*width, 3*height), interpolation = cv2.INTER_CUBIC)
  fig = plt.gcf()
  fig.set_size_inches(18, 10)
  plt.axis("off")
  plt.imshow(cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB))
  plt.show()
```

Here's the Darknet command for YOLOv4 object detection.

```
#!./darknet detector test <path to .data file> <path to config> <path to weights> <path to image>
```

```
#test out our detector!
!./darknet detect /content/darknet/custom-yolov4-detector.cfg yolov4.weights /content/New-Delhi 2.jpg -dont-show
```

```
# Run the following code to show the prediction image, which YOLO automatically stores (as 'predictions.jpg') with bounding boxes for detected items.
```

```
imShow('predictions.jpg')
```